Comparación de Métodos en la Multiplicación de Matrices Cuadradas con y sin OpenMP

Oscar Correcha, Oscar Corzo

***Abstract*—Este artículo presenta una comparación meticulosa con el objetivo de demostrar la reducción de tiempo lograda mediante el uso del estándar OpenMP. El objetivo principal del experimento fue abordar las siguientes preguntas: ¿Existe una diferencia significativa en el tiempo de ejecución entre un algoritmo implementado en C con OpenMP y su contraparte sin el uso de OpenMP? y ¿Hay una disparidad discernible en el tiempo de ejecución entre algoritmos implementados en C y aquellos implementados en Python (utilizando la biblioteca NumPy) y en R?.**

El experimento se llevó a cabo en tres computadoras diferentes. Los resultados revelan una influencia sustancial en el tiempo de ejecución al utilizar el estándar OpenMP. Las conclusiones no solo confirman una diferencia significativa en el tiempo de ejecución entre algoritmos en C con y sin OpenMP, sino que también resaltan las disparidades en el tiempo de ejecución entre algoritmos implementados en C y aquellos en Python (utilizando la biblioteca NumPy) y en R. Este estudio aporta conocimientos valiosos sobre las implicaciones prácticas de aprovechar Open MP para la eficiencia computacional, proporcionando una base para la toma de decisiones informada en la implementación de algoritmos en diferentes lenguajes de programación.

***Index Terms*—Open MP, Paralelo, Estadística, ANOVA, Regresion Lineal, Python, R, C.**

I. Introducción

La creciente complejidad de los problemas abordados por los investigadores, así como la proliferación de conjuntos de datos masivos, han elevado la importancia de lograr tiempos de ejecución óptimos. La capacidad de reducir significativamente el tiempo necesario para ejecutar algoritmos puede tener un impacto directo en la eficiencia operativa y la viabilidad de las soluciones propuestas. En este contexto, el presente informe se enfoca en la evaluación y comparación meticulosa de los tiempos de ejecución de algoritmos implementados con y sin el estándar de programación en paralelo Open MP principalmente en el lenguaje C. El énfasis recae en el papel crítico que desempeñan estos tiempos en la mejora continua de la eficiencia de los algoritmos, proporcionando a los investigadores una base sólida para la toma de decisiones informada en la implementación de soluciones computacionales avanzadas y eficientes.

II. Proceso o sistema que se estudia y variables respuestas de interés

El proceso estudiado es la velocidad de ejecución de código escrito en C usando y no usando Open MP, adicionalmente se compararán estas salidas con ejecución de código escrito en Python y R.

A. Definiciones básicas

Open MP es una interfaz de programación de aplicaciones para la programación multiproceso de memoria compartida en múltiples plataformas. Esta interfaz ofrece un estándar para programar aplicaciones paralelas en sistemas de memoria compartida, portable, que permite paralelismo incremental, independiente del software que lo soporte.

Adicionalmente, esta interfaz ofrece directivas de compilador, ofrece unas pocas funciones de librería, variables de entorno, entre otras cosas.

En pocas palabras, partiendo de un programa serial se puede obtener un programa paralelo con Open MP añadiendo directivas que especifiquen una región paralela, o un reparto de tareas o una sincronización entre hilos ejecutados por cada uno de los procesadores de la máquina. [1]

La ley de los grandes números es un teorema fundamental de la teoría de la probabilidad que indica que, si se repite un número de veces considerable un experimento, la frecuencia de que suceda un determinado suceso tiende a ser una constante. [2]

El **Análisis de varianza (ANOVA)** de un factor es un método estadístico para examinar las diferencias en las medidas de 2 o más grupos. Usualmente, el ANOVA de un factor se emplea cuando tenemos una única variable o factor independiente y el objetivo es investigar si las variaciones o diferentes niveles de ese factor tienen un efecto medible sobre una variable dependiente. [3]

B. Lenguajes de programación utilizados

- C
- Python (Numpy)
- R

C. Variables que intervienen en el experimento

- Computadoras
- Lenguajes de programación
- Algoritmos

La variable respuesta corresponde al tiempo medido en segundos de una multiplicación de matrices $N \ x \ N$ con valores entre 1 y 100 generadas aleatoriamente, estas ejecuciones se desarrollaron una vez reiniciada la computadora, desconectando el internet y cualquier evento, esperando a que todos los procesadores (CPU) se encuentren en estado de reposo, aproximadamente al 100% de inactividad.

III. Objetivo del experimento

En la presente investigación, se lleva a cabo una meticulosa comparación con el objetivo de evidenciar la existencia de reducción de tiempo mediante el empleo del estándar Open MP. El experimento se concibe con el propósito de responder a las siguientes interrogantes:

1) ¿Se constata una diferencia significativa en el tiempo de ejecución entre un algoritmo implementado en C con Open MP y su contraparte sin la utilización de Open MP?
2) ¿Se aprecia una disparidad en el tiempo de ejecución entre los algoritmos implementados en C y aquellos implementados en Python (utilizando la biblioteca Numpy) y en R?

Es imperativo subrayar la imposibilidad de generalizar los resultados obtenidos a otros lenguajes de programación, dado que cada uno puede diferir sustancialmente en su modalidad de ejecución. Asimismo, es crucial reconocer que las limitaciones inherentes a los equipos utilizados impiden replicar de manera exacta las condiciones usuales en las cuales estos algoritmos son ejecutados. Las conclusiones derivadas de este estudio únicamente pueden ser interpretadas dentro del contexto específico de las condiciones experimentales establecidas, aunque podrían proporcionar indicios sobre los resultados que se podrían obtener bajo condiciones más convencionales.

IV. Identificación de todas las fuentes de variación

A. Factores de estudio y sus niveles

Los factores de estudio en la presente investigación se han estructurado cuidadosamente para evaluar la eficiencia de distintos tipos de algoritmos bajo condiciones representativas. El factor de tratamiento denominado ”tipo de algoritmo” incorpora tres lenguajes de programación: C, C con Open MP, Python (Numpy) y R. Cabe destacar que, en el marco de este estudio, no se ejerce control sobre la presencia de daemons o tareas en las computadoras utilizadas; se asume que dichas computadoras son representativas de las configuraciones comúnmente disponibles en el mercado.

Otro factor de tratamiento relevante es el ”tamaño de matriz”, en el cual se han seleccionado cinco dimensiones: 100x100, 200x200, 300x300, 400x400 y 500x500, con valores aleatorios comprendidos entre 1 y 100. Es imperativo señalar que la validez general de los resultados dependerá de la presunción de que las computadoras empleadas reflejan de manera adecuada la diversidad de configuraciones presentes en la población general. Con el fin de obtener una representación robusta y significativa, se medirá el tiempo de ejecución de cada algoritmo en 30 repeticiones, con el propósito de minimizar la variabilidad y garantizar la fiabilidad de las conclusiones extraídas del experimento.

B. Factores de bloque, de ruido y covariables

En adición a las disparidades en la ejecución del algoritmo en distintos lenguajes de programación, se reconocen otras fuentes potenciales de variabilidad que incluyen las diferencias en los procesadores utilizados y las condiciones ambientales, como el calor generado por las computadoras y las fluctuaciones en la temperatura diaria a su vez vinculadas a la fecha de fabricación. Aunque estas variables podrían haberse considerado para la aplicación de técnicas de bloqueo o como covariables, el investigador evaluó que las unidades experimentales no demostraban la suficiente variabilidad como para justificar el bloqueo. En lugar de esto, se optó por medir los cambios temporales, específicamente, la diferencia entre el tiempo final y el tiempo inicial de ejecución.

Otras fuentes de variación, como la generación de números aleatorios y la impresión en pantalla, fueron identificadas, aunque se consideraron de importancia menor. No se incorporó ningún factor de ruido al diseño experimental. Este enfoque se adoptó con la convicción de que el control de estas variables era suficiente para mantener la integridad del experimento y garantizar la interpretación adecuada de los resultados.

C. Unidades experimentales

El experimento se llevará a cabo utilizando distintas computadoras, a las cuales se les desactivará las conexiones de red, bluetooth y cualquier otro tipo de conexión inalámbrica. Para asegurar condiciones óptimas, se reiniciarán las computadoras, y se verificará que las unidades centrales de procesamiento (CPU) se encuentren en estado de reposo, aproximadamente al 100% de inactividad, antes de la ejecución de los algoritmos a través de la consola. Durante esta ejecución, se registrará el tiempo de ejecución de cada algoritmo en segundos.

Cabe destacar que esta metodología se basa en los resultados de un estudio piloto que indicó que este enfoque de ejecución es suficiente para abordar cualquier variación potencial en los tiempos de ejecución de los algoritmos en función de los tamaños de las matrices. Este procedimiento se seleccionó con la confianza de que proporcionará una evaluación rigurosa y consistente de la eficiencia de los algoritmos bajo estudio, minimizando las posibles interferencias externas y garantizando la validez de las conclusiones extraídas.

V. Regla de asignación de las unidades experimentales a los tratamientos

Se realizará un número equitativo de observaciones en cada nivel de los factores considerados. En consecuencia, se llevarán a cabo n ejecuciones para cada combinación de algoritmo y tamaño de matriz. La elección de n se estableció en 30 para cumplir con los requisitos del Teorema del Límite Central (**TLC**). Por motivos de simplicidad en la ejecución, no se implementará la aleatorización en este proceso experimental.

VI. Procedimiento experimental, mediciones y dificultades anticipadas

Los algoritmos son compartidos por GitHub para la reproducibilidad del experimento, al verificar que los CPU’s se encuentren en estado de reposo, aproximadamente al 100% de inactividad. Después que se han ejecutado los algoritmos, en pantalla se muestra el tiempo de ejecución en segundos. Estos tiempos serán registrados. El análisis se realizará sobre las diferencias en el uso de Open MP.

A. Dificultades experimentales esperadas

- Tareas programas, o deamons en la computadora. Tambien conexiones inalámbricas activas. Por tanto, los datos pueden no mostrar diferencias en los tiempos.
- Tiempos afectados por la ejecución anterior, correlación entre muestras.
- Tiempos mayores a los esperados en el estudio piloto.

B. Dificultades encontradas durante la experimentación

- En el estudio piloto, se observaron tareas que no fueron desactivadas y que impactaron en el tiempo, como la conexión a Internet o la activación del bluetooth en los audífonos.
- El uso de consola disminuía los tiempos, usar herramientas IDE aumentaba tiempos.
- En el estudio piloto se evidenciaron correlaciones entre los tiempos.

VII. Estudio piloto

Se corrió un estudio piloto con dos propósitos: Primero, para identificar las dificultades listadas en la Sección 5, segundo, para obtener una estimación de la varianza para determinar el número de réplicas. La varianza del error fue estimada aproximadamente de 0.146, el valor del MSE en dicho estudio piloto. De hecho, este es una sobre estimación y hubiese sido mejor usar un intervalo de confianza unilateral para la varianza.

VIII. Especificación del modelo estadístico, análisis o pruebas a realizar y tamaños de muestra

Dado que se tendrá cuidado en controlar todas las fuentes extrañas de variación, se asumió que el siguiente modelo es una aproximación razonable:

$$Y_{ij} = \mu + \ \alpha_i + \ \varepsilon_{ij}, \quad \varepsilon_{ij} \overset{iid}{\sim} N\left(0,\ \sigma^2\right), \quad i, j = 2$$

$$\sum_{i=\ 1}^{2} \alpha_i = 0$$

Con el fin de responder a la pregunta sobre las diferencias en los tiempos, una ANOVA de un factor de efectos fijos será calculada al 0.05 de significancia para probar:

$$H_0 :\ \alpha_1 = \alpha_2 \ = 0$$

$$H_{\mathbf{1}} :\ algún\ \alpha_{\mathbf{i}}\ diferente\ de\ 0$$

Donde el estadístico de prueba y su distribución es normal y el criterio de decisión con valor P es 0.05. Si se detecta significancia estadística, se estimarán las medias y los efectos según tipo algoritmo. Para hallar mejor las diferencias entre pares de tratamientos, se construirán el test de Tukey con una significacia del 95%.

TABLE I
ANOVA FxC

Experimentación	Estadísticos	Valores P	Decisión
100	161.445861	2.10e-18	significativo
200	181.846528	1.57e-19	significativo
300	5983.452207	4.963132e-60	significativo
400	974.006617	5.93e-38	significativo
500	20827.584568	7.63e-76	significativo

A. Cálculo del número de observaciones necesarias

Se determinó un tamaño de muestra de n=30 para cada algoritmo y tamaño de matriz, con el objetivo de identificar una disminución significativa en los tiempos, de al menos un 8%, entre los algoritmos implementados en C con OpenMP y aquellos sin OpenMP. Este análisis se llevó a cabo con una confianza del 90% y un nivel de significancia establecido en 0.05.

B. Revisión de las decisiones anteriores

No resulta arduo obtener treinta observaciones de cada tratamiento; los ajustes menores al procedimiento experimental, que se consideraron necesarios durante la studio piloto, ya han sido implementados.

IX. Datos recolectados y análisis estadístico de resultados

A. Características de las máquinas donde se ejecutaron las pruebas

1) Máquina personal: Esta máquina cuenta con 2 procesadores Intel core I7, sistema operativo Windows 10 con virtualización de WSL2 - Ubuntu versión 22.04.3 LTS.

2) Cliente de Condor.javeriana.edu.co: Esta máquina posee una arquitectura x86-64 y tiene un total de 2 procesadores, ambos agrupados dentro de un solo socket.

3) Máquina personal: Esta máquina cuenta con 3 procesadores Intel core I5, virtual Oracle Virtual Box - Ubuntu versión 22.04.3 LTS, los cuales no se encuentran limitados en términos de capacidad. Respecto a su placa base, esta posee una memoria de 4714 MB. Consiste en un virtualizador basado en innotek GmbH VirtualBox.

B. Información recolectada

Los datos en formato csv se encuentran alojados en el repositorio.

De la figura 1 y la figura 2 se aprecia una diferencia visual entre las medias de los tiempos medidos por las ejecuciones de los algoritmos FxC y FxF para los lenguajes C, Python (con Numpy) y R.

De la figura 3 se puede apreciar una posible diferencia significativa entre las medías del tiempo medido por ejecuciones con OpenMP y aquellas sin OpenMP, lo cual es corroborado por los resultados de la prueba ANOVA en la Tabla 1 para el algoritmo FxC ya que ejecuciones con OpecMP (línea azul) permanece por debajo en todos los experimentos, adicionalmente en la figura 4 la dispersión es menor que ejecuciones sin OpenMP (línea naranja). En la prueba de

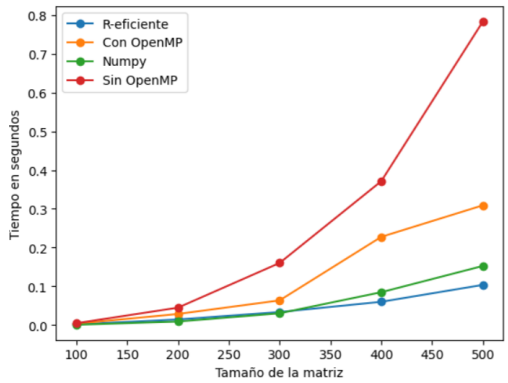

Fig. 1. Tiempo de ejecución de la multiplicación matricial para el algoritmo FxC en el PC 1

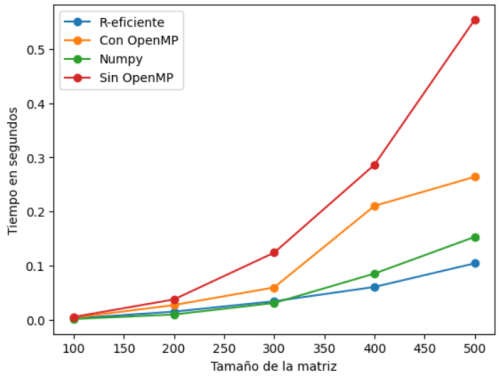

Fig. 2. Tiempo de ejecución de la multiplicación matricial para el algoritmo FxF en el PC 1

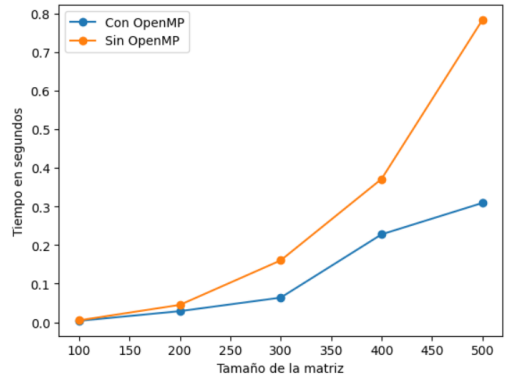

Fig. 3. Tiempo promedio de ejecución del algoritmo FxC para el lenguaje C

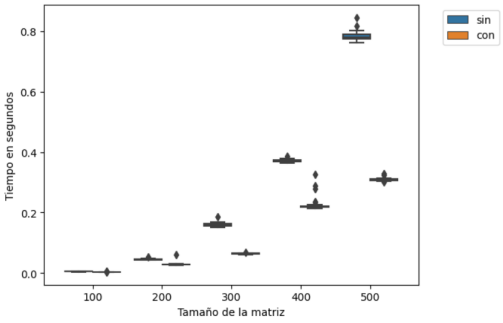

Fig. 4. Diagrama de cajas y bigotes de los datos recopilados al ejecutar el algoritmo FxC usando lenguaje C

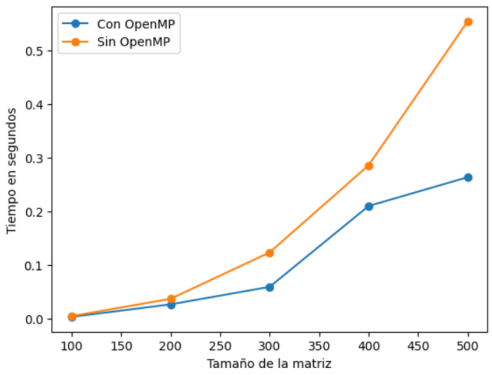

Fig. 5. Tiempo promedio de ejecución del algoritmo FxF para el lenguaje C

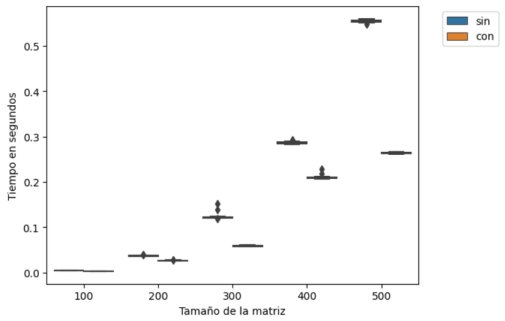

Fig. 6. Diagrama de cajas y bigotes de los datos recopilados al ejecutar el algoritmo FxF usando lenguaje C

ANOVA, con una significancia del 5% usando el algoritmo FxC, se rechazan todas las pruebas de hipótesis por lo que podemos inferir que existe una diferencia entre usar OpenMP y no usarlo.

De la figura 5, como ocurre en la figura 3, con una significancia del 5% para el algoritmo FxF, existe evidencia muestral significativa que rechazan todas las pruebas de hipótesis, por lo que podemos volver a inferir que existe una diferencia entre ejecuciones con OpenMP y aquellas sin OpenMP. Por otro lado, la figura 6 denota un tiempo de ejecución aún mas consistente para este algoritmo que para el algoritmo de FxC presentado en la figura 4 puesto que se aprecia que la varianza de los tiempos es mucho menor.

De la tabla IV se observa que los coeficientes para el algoritmo FxF son más pequeños eso indica que el consumo de tiempo al hacer más grande la matriz tomara menos tiempo que usar el algoritmo FxC. La regresión para el algoritmo FxF tiene un ajuste 3.3% mayor que el algoritmo FxC, que, aunque podría ser despreciable el error es mucho menor para el algoritmo FxF lo que indica que tiene menos dispersión y

TABLE II
ANOVA FxF

Experimentación	Estadísticos	Valores P	Decisión
100	692.124770	6.24e-34	significativo
200	3413.493849	3.06e-53	significativo
300	2989.318783	1.34e-51	significativo
400	8518.222782	1.24e-64	significativo
500	335453.914509	8.17e-111	significativo

TABLE III
ANOVA FxF

Experimentación	Estadísticos	Valores P	Decisión
100	692.124770	6.24e-34	significativo
200	3413.493849	3.06e-53	significativo
300	2989.318783	1.34e-51	significativo
400	8518.222782	1.24e-64	significativo
500	335453.914509	8.17e-111	significativo

TABLE IV
REGRESIÓN

Parámetro	Valor FxC	Valor FxF
Intercepto	-0.29190067	-0.11627441
Pendiente	0.00188289	0.00080968
R2	87.1%	90.4%
MSE	0.0104	0.0014

más control en la toma de muestras.

La figura 7 es el resultado gráfico de la prueba de Tuckey, (el resultado numérico se encuentra en el código) con una significancia del 5% usando el algoritmo filas por columnas contra el algoritmo filas por filas, se rechaza las pruebas para las matrices igual o mayores a 300x300, lo que nos indica que existe una reducción significativa en los tiempos para la ejecución del algoritmo con Open MP FxF, sin embargo en el análisis anterior, se encuentra una diferencia pequeña para las matrices iguales o menores a 200x200 que para el algoritmo puede ser despreciable.

Para continuar con la segunda parte del experimento se ejecutaron únicamente los algoritmos de FxC y FxF en lenguaje C con y sin Open MP en las 2 máquinas restantes mencionadas con anterioridad. El promedio de tiempos para los experimentos del PC 2 pueden verse en la figura 8 y el promedio de tiempos de los experimentos del PC 3 se aprecian en la figura 9.

Los resultados obtenidos respaldan la afirmación de que el algoritmo FxF es considerablemente más eficiente que su contraparte FxC. La disparidad observada durante la ejecución en la máquina 1 revela que, sorprendentemente, el código implementado con Open MP para ambos algoritmos resultó ser menos eficiente en términos generales.

Este resultado inesperado podría atribuirse a diversas circunstancias. En relación con la máquina cliente del cluster condor.javeriana.edu.co, la posibilidad de acceso simultáneo por parte de varios usuarios podría haber evitado que los

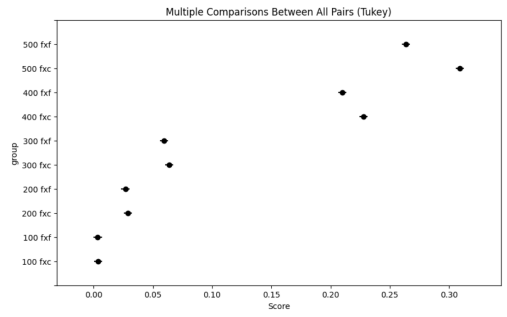

Fig. 7. Distribución de la respuesta según Open MP FxF y FxC

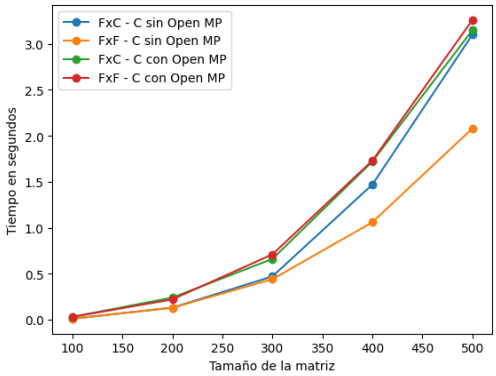

Fig. 8. Tiempo promedio de ejecución de los algoritmos FxC y FxF para el lenguaje C en la máquina cliente de Condor (PC 2)

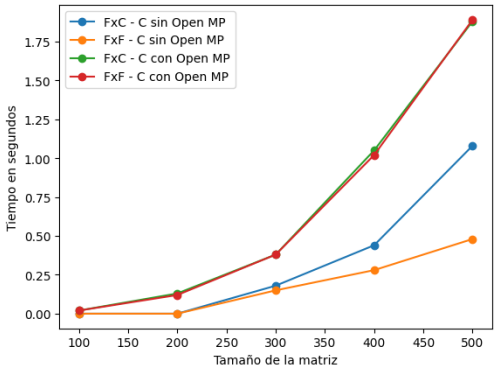

Fig. 9. Tiempo promedio de ejecución de los algoritmos FxC y FxF para el lenguaje C en la máquina virtual (PC 3)

procesadores alcanzaran el 100% de su estado ocioso al ejecutar los algoritmos. Su respectivo diagrama de cajas y bigotes (ver figura 10) denota inmediatamente una varianza muestral mayor a la obtenida en el experimento del PC 1.

Por otro lado, en cuanto a la máquina virtual, el rendimiento se vio afectado principalmente por la interfaz gráfica utilizada para la ejecución de los algoritmos. La conexión a esta interfaz impedía que los tres procesadores alcanzaran su máximo nivel de inactividad mientras estuvieran conectados quienes ejecutaron los algoritmos. Esto lo podemos apreciar en el

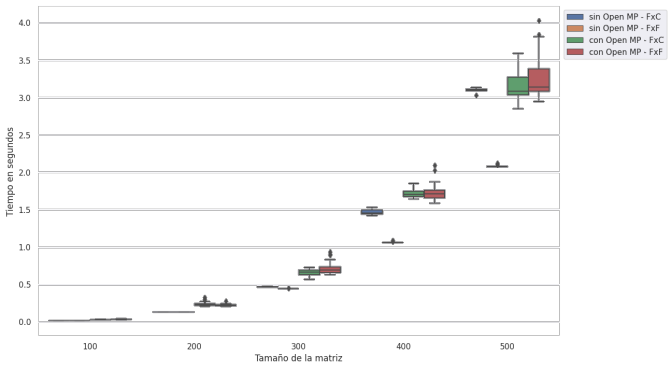

Fig. 10. Diagrama de cajas y bigotes para la salida de los algoritmos FxC y FxF en lenguaje C para la máquina cliente de condor (PC 2)

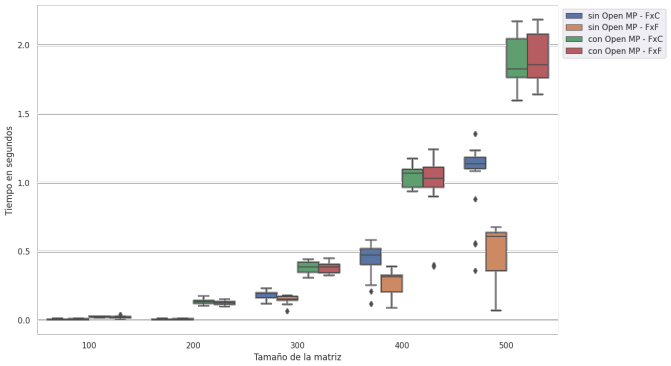

Fig. 11. Diagrama de cajas y bigotes para la salida de los algoritmos FxC y FxF en lenguaje C para la máquina virtual (PC 3)

diagrama de cajas y bigotes de la figura 11 donde, para matrices con N mayor o igual a 400 los datos de salida presentan una varianza inclusive mayor que la varianza de los experimentos del PC 2, que a su vez ya era superior a la varianza de los experimentos realizados en el PC 1.

X. Conclusiones y recomendaciones

Con base en los resultados experimentales, los principales hallazgos fueron los siguientes: Usando Python usando algoritmo FxC o FxF es el lenguaje más lento, lo sigue R usando el usando algoritmo FxC y FxF. Numpy es más eficiente que usar los anteriormente nombrados ya que implementa Open MP. Open MP para matrices pequeñas no tiene una gran ventaja, sin embargo, para matrices mayores a 200x200 es más eficiente que no usar Open MP.La multiplicación básica de matrices en R es más efciente que los demás algoritmos y adicionalmente no usa Open MP, solamente usa un CPU. Adicionalmente se comprobó que el algoritmo de FxF es más rápido que el de FxC. Esto puede deberse a que en ambas matrices que se multiplican se accede a los elementos por fila. Este patron de acceso tiende a ser más eficiente en términos de caché, ya que los elementos accedidos están más cerca entre si en la memoria, lo que puede resultar en un mejor rendimiento. Como recomendación para mejorar este experimento puede considerarse lo siguiente: Si el tiempo de ejecución no es importante para el investigador o la matriz es pequeña se recomienda usar R o C sin usar Open MP ya que utiliza mucho tiempo escribir el código en dicho formato y en R solo es necesario 1 instrucción para dicha ejecución. Si el tiempo de ejecución es importante para el investigador o las matrices son próximas o mayores a 300x300 se recomendaría revisar la posibilidad de escribir dicho código usando C con Open MP o utilizar numpy o R e implementar el código de FxF.

Referencias

[1] Mg. Javier Echaiz. *OPEN MP Sistemas operativos y distribuidos*. URL: https://cs.uns.edu.ar/~gd/soyd/clases/13-OpenMP.pdf. (accessed: 13.11.2023).

[2] Paula Nicole Roldán. *Ley de los grandes números*. URL: https://economipedia.com/definiciones/ley-los-grandes-numeros.html. (accessed: 13.11.2023).

[3] S.A. *ANOVA de un factor*. URL: https://www.jmp.com/es_co/statistics-knowledge-portal/one-way-anova.html. (accessed: 13.11.2023).